REMARKS

This amendment is responsive to the Office Action[1] mailed on June 30, 2005. Claims 1-89 were presented for examination and were rejected. Claims 37, 49, 57-58 and 68 are amended. No claims are added or canceled. Thus, claims 1-89 are pending. Claims 1, 12, 24, 37, 49, 52, 59, 63, 68, 70, 75 and 81 are independent claims.

Preliminarily, in the Office Action, Applicants were not able to locate a rejection for claims 35 and 36. These claims are dependent from independent claim 24 which was rejected under 35 U.S.C. § 103(a) as being un-patentable over U.S. Patent No. 5,960,200 to Eager et al. (hereinafter "Eager") in view of U.S. Patent No. 5,960,200 to Bapat (hereinafter "Bapat"). For purposes of responding herein, Applicants assume that the Examiner would have rejected claims 35-36 over Eager in view of Bapat in further view of admitted prior art based on other similar rejections provided in the Office Action.

Furthermore, Applicants shall respond to the Office Action generally in the order in which issues are presented. After responding to the 35 U.S.C. § 101 rejection, Applicants shall next respond to the 35 U.S.C. § 103(a) rejections as a group, to the extent possible, and treat the 35 U.S.C. § 102(b) rejections separately, thereafter, generally following a sequence similar to that provided in the Office Action.

35 U.S.C. § 101 REJECTION:

Claims 37, 49 and 68 were rejected under 35 U.S.C. § 101 because the claimed

[1] The Office Action may contain a number of statements characterizing the cited references and/or the claims which Applicants may not expressly identify herein. Regardless of whether or not any such statement is identified herein, Applicants do not automatically subscribe to, or acquiesce in, any such statement. Further, silence with regard to rejection of a dependent claim, when such claim depends, directly or indirectly, from an independent claim which Applicant(s) deems allowable for reasons provided herein, is not acquiescence to such rejection of that dependent claim, but is recognition by Applicant(s) that such previously lodged rejection is moot based on remarks and/or amendments presented herein relative to that independent claim.

invention was allegedly directed to non-statutory subject matter. The Examiner suggested amending each of the claims to embody the computer program on a computer readable medium, and Applicants have complied therewith. Applicants respectfully request that this rejection be withdrawn.

35 U.S.C § 103(a) REJECTIONS:

Claims 1, 2, 4-6, 8-10, 12, 24-25, 27-29, 31-34, 37-39, 41-42, 44-47, 49-50, 52-55, 57-59, 60-62 and 75-78 are rejected under 35 U.S.C. § 103(a) as being un-patentable over Eager in view of Bapat.

On page 16 of the Office Action, claims 63-67 and claims 68-69 are also rejected under 35 U.S.C. § 103(a) as being un-patentable over Eager in view of Bapat.

Claims 3, 7, 11, 14, 18, 23, 26, 30, 40, 43, 48, 51, 56 and 79-80 are rejected under 35 U.S.C. § 103(a) as being un-patentable over Eager in view of Bapat and further in view of Applicants admitted prior art[2] (hereinafter "APA").

Claims 74 and 84-88 are rejected under 35 U.S.C. § 103(a) as being un-patentable over Eager in view of APA.

Applicants respectfully traverse these rejections, primarily for two reasons: (1) Eager's header files are not equivalent to Applicants' claim-recited header files and/or (2) Eager does not teach or imply any claim-recited subject matter equivalent to Applicants' blocks 302 and 303 in Fig. 3, as explained in detail below:

With the exception of independent claims 63 and 68, each independent claim in

[2] Applicants take issue with this phraseology. As expressed in a prior response filed on August 2, 2004, page 29, "Accordingly, although Applicants adopt herein the Examiner's phraseology 'admitted prior art' in reference to both Figs. 2A/2/B and schema included therein for purposes of facilitating communication on the record, Applicants do not necessarily agree that Figs. 2A/2B are prior art vis-à-vis the claimed subject matter."

this category of rejections, namely, claims 1, 12, 24, 37, 49, 52, 59 and 75, recites "header files." Applicants' header files are discussed in their specification and are shown at least in Fig. 3. Applicants' header files are derived from their legacy software and are an INPUT to their translator 300 function. By contrast, Eager's header files are something else entirely. Consider, for example, claim 1:

> A computer system employing management software written in a first computer language compatible with first software architecture and not compatible with second software architecture, said system comprising: a *schema formed within said first software architecture; header files contained within said schema, said header files being represented in said first language* and capable of being utilized by said management software; means for manipulating said header files to locate public functions and/or data attributes of said header files; means, responsive to operation of said manipulating means, for emitting code that calls said public functions and/or data attributes in said first language to obtain called public functions and/or data attributes; and, means for converting said called public functions and/or data attributes to representations of said called public functions and/or data attributes formed in a different computer language compatible with said second software architecture. (Emphasis added.)

It is clear that Applicants' header files are contained within the schema which is formed within the first software architecture. But, Applicants' first software architecture is legacy architecture with which the recited "management software written in a first computer language" is compatible. This is clear from the dependent claims. For example, claim 3, dependent from claim 1, defines the first computer language as RAID++ and the different computer language as XML/CIM. From Applicants' Fig. 2A, for example, it is clear that RAID++ is legacy architecture and XML is the new, standard architecture.

The Office Action, page 4, cites Eager, column 29, lines 11-15 to show "header files contained within said schema". Indeed, this section refers to a schema header file 238d, which is shown in Fig. 25, in Target DDL (Data Definition Language) 238. But

this header file is located AFTER conversion in DDL Converter 235 which occurs within DDL Conversion Utility 230 which, in turn, functions within Eager's re-architecting system 20, as shown in Fig. 1 of the reference. Thus, Eager's "header file" is created by the process of conversion WITHIN its legacy-to-non-legacy conversion system 20: "The first converter 235a also *creates* the schema header file 238d....." (Eager, col. 29, lines 11-12, emphasis added.) By contrast, Applicants' header was previously in existence in the legacy system as shown, e.g., in Applicants' Fig. 3, block 100, well prior to being supplied to its conversion system.

In claim 1, because Applicants' schema is "formed within said first software architecture" (within legacy architecture), as recited, and because Applicants' header files are "contained within said schema" (thus contained within legacy architecture), as recited, Eager's disclosure of the appearance of header files *after* conversion from its legacy architecture cannot disclose or suggest Applicants' header files. In other words, the recited claim element: "header files contained within said schema, said header files being represented in said first language and capable of being utilized by said management software" is not disclosed or suggested by Eager because the schema in which Eager's header files appear is not remotely equivalent to the schema in which Applicants' header files appear. Applicants' header files were previously in existence well prior to being supplied to its conversion system, *before* the conversion process but Eager's header files are created *after* the conversion process or, at best, *during* the conversion process. Bapat and/or APA do not cure this deficiency in Eager. For this reason alone, the rejection of claim 1 under 35 U.S.C. § 103(a) over Eager in view of Bapat should be withdrawn and the claim allowed.

Furthermore, the Office Action states that Eager does not disclose Applicants' (1) "means for manipulating said header files to locate public functions and/or data attributes of said header files" nor (2) "means, responsive to operation of said manipulating means, for emitting code that calls said public functions and/or data attributes in said first language to obtain called public functions and/or data attributes" as recited in claim 1. (Office Action, page 5) Applicants agree.

The Office Action relies on Bapat to disclose these two claim elements. Applicants respectfully disagree. Even if the sections relied-upon in Bapat in the Office Action did disclose subject matter arguably equivalent to bare language of these two claim elements (which they do not for reasons presented below) since Bapat does not cure the above-noted deficiency of Eager, any such disclosure in Bapat in combination with Eager is also inherently deficient with respect to claim 1. In other words, any alleged manipulation of header files which may appear in Bapat when combined with Eager would provide, at best, a manipulation of *irrelevant* Eager header files with respect to claim 1. And, any means, responsive to this manipulation of these header files, for emitting code that calls public functions, etc. when combined with Eager would provide, at best, means for emitting code in response to manipulation of *irrelevant* header files.

Therefore, the combination of Eager with Bapat does not disclose or suggest: "means for manipulating said header files to locate public functions and/or data attributes of said header files; means, responsive to operation of said manipulating means, for emitting code that calls said public functions and/or data attributes in said first language to obtain called public functions and/or data attributes", as recited in claim 1, because the

recited header files which are contained within the schema which is formed within the first, legacy architecture are not shown or implied in the combination of Eager and Bapat.

As noted above, the sections in Bapat upon which the Office Action relies do not support the position taken in the Office Action in the first place. For example, the Office Action relies upon column 3, lines 53-54 in Bapat to show "means for manipulating...". However, looking at column 3 lines 49-54, including the cited section, it says:

> A *program* is stored in the computer memory means for representing *DATA* in an object class hierarchy. An access method is stored in the computer memory and is responsive to the *program* for *manipulating* a representation of the object class hierarchy stored in a relational table schema. (Emphasis added.)

Although this section mentions the term "manipulating", it is not otherwise relevant. It says that a program is stored for representing *DATA* and that an access method is *responsive to that program* for the purpose of manipulating a representation of an object class hierarchy. Thus, whatever this manipulation involves, it is responsive to a program that represents data. But, a header is not data, per se; it is a notification or declaration about its associated data.

The Office Action then combines this section with another section that mentions the term "header." "Next, at step 174, the indicated C++ source and header files are opened." (Bapat, column 9, lines 1-2). Combining these two sections, one may logically conclude that either (1) any manipulation is being made upon only the C++ source files (data), or (2) if manipulation is also being made upon header files in Bapat, those files may contain data if manipulation is based on a program stored for representing data. If the former, that manipulation is irrelevant to Applicants' recited "means for manipulating said header files..." since C++ source files are not header files. If the latter, that

manipulation is also irrelevant since Bapat's header files would then contain data, but Applicants' header files are comprised of other than data. This is clear from at least Applicants' Fig. 3, legacy architecture 100, where RAID++ represents data and its associated HEADER FILES are other than data. *See* the specification, at least page 16, lines 7-8.

Moreover, it is respectfully submitted that the reliance in the Office Action, page 5, upon column 6, lines 28-29 in Bapat: "it is desirable to be able to provide persistent storage of the attributes of these objects managed by the network management system." to show Applicants' recited "to locate public functions and/or data attributes of said header files" is without merit. Persistent storage of attributes of objects is not equivalent to public functions and/or data attributes of header files.

Bapat is thus ineffective as a reference on two levels. First, it doesn't cure the wrong header file deficiency of Eager where its combination with Eager would not disclose or suggest Applicants' claimed subject matter in any event. Second, the citations within Bapat, in and of themselves, are irrelevant when compared with the bare claim language against which they are being applied, regardless of the non-equivalence of Eager's headers and Applicants' headers.

For these additional reasons, it is respectfully requested that the rejection of independent claim 1 be withdrawn and the claim allowed.

Independent claims 12, 24, 37, 49 and 52 each recite header subject matter similar to that recited in claim 1, as follows, (with emphasis added):

Claim 12 recites, *interalia*: "A computer network employing a computer system utilizing *management software written in a first computer language compatible with first*

software architecture and not compatible with second software architecture, said network comprising: a *schema formed within said first software architecture; header files contained within said schema*, said header files being represented in said first language and capable of being utilized by said management software."

Claim 24 recites, *interalia*: "A method for utilizing standardized software architecture to be practiced in a computer system employing *management software written in a first computer language compatible with first software architecture and not compatible with said standardized software architecture*, said method comprising: *said management software utilizing a schema having header files in said first language*."

Claim 37 recites, *interalia*: "A computer program product including *management software written in a first language* and embodied in a computer-readable medium *for operation on a computer system designed in accordance with first software architecture and not compatible with other than said first software architecture*, said computer program product comprising: programmable code for utilizing *a schema having header files in said first language*."

Claim 49 recites, *interalia*: "A computer program product compatible with preferred non-legacy software architectures and operating in a computer system employing management software written in *a first computer language compatible with legacy software architecture* and not compatible with said preferred non-legacy software architectures, said computer program product embodied in a computer readable medium and comprising: programmable code for utilizing *a schema having header files in said first language*."

Claim 52 recites, *interalia*: "In a computer network including a computer system having a functional system therein with management software including *a schema for managing said functional system under control of said computer system in accordance with first software architecture*, a translator-compiler for permitting communication about said managing said functional system to be transmitted between said computer system and computer devices operating under second software architecture, said translator-compiler comprising: program code for accessing *header files within said schema* to obtain a header file containing particular information."

Claim 59 recites, interalia: "In a computer network including a computer system and a functional system controlled by said computer system, *management software compatible with legacy software architecture having header files*, said management software being deployed on both said computer system and said functional system, said *management software comprising: translator software means for receiving and manipulating said header files*."

As can be seen, in each of these independent claims, the header files are included in a schema in a first language and/or in a first architecture (legacy architecture). As such, they are not equivalent to the header files in Eager which have been shown above to result from Eager's process of conversion from enterprise (legacy) to distributed (non-legacy) architectures. It is respectfully submitted that the 35 U.S.C.§ 103(a) rejection of claims 12, 24, 37, 49, 52 and 59 should be withdrawn and the claims allowed, at least for the reasons given above with respect to claim 1.

Claims 2-11, dependent from claim 1, are allowable, at least for reasons based on their dependency, directly or indirectly, from allowable base claim 1.

Claims 13-23, dependent from claim 12, are allowable, at least for reasons based on their dependency, directly or indirectly, from allowable base claim 12.

Claims 25-36, dependent from claim 24, are allowable, at least for reasons based on their dependency, directly or indirectly, from allowable base claim 24.

Claims 38-48, dependent from claim 37, are allowable, at least for reasons based on their dependency, directly or indirectly, from allowable base claim 37.

Claims 50-51, dependent from claim 49, are allowable, at least for reasons based on their dependency, directly or indirectly, directly or indirectly, from allowable base claim 49.

Claims 53-58, dependent from claim 52, are allowable, at least for reasons based on their dependency, directly or indirectly, from allowable base claim 52.

Claims 60-62, dependent from claim 59, are allowable, at least for reasons based on their dependency, directly or indirectly, from allowable base claim 59.

Although independent claim 75, rejected under 35 U.S.C. § 103(a) over Eager in view of Bapat, also recites header files which are, in fact, the same header files as those in the other claims, and although Applicants submit that claim 75 is allowable for the same reasons set forth above with respect to claim 1, there are additional reasons why claim 75 is allowable. These additional reasons are also applicable to independent claims 63 and 68, also rejected under 35 U.S.C. § 103(a) over Eager in view of Bapat, but which do not include recitations of headers.

Consider, first, independent claim 75:

> In a computer network including a computer system and a storage system controlled by said computer system, a method for managing storage compatible

> with software architecture having header files, said method being deployed on both said computer system and said storage system, said method comprising: translating and manipulating said header files to obtain translated and manipulated header files; *receiving first requests from outside of said network in first language incompatible with said software architecture*; in cooperation with said translated and manipulated header files, *obtaining responses to said first requests in second language compatible with said software architecture*; and, in cooperation with said translated and manipulated header files, *converting said responses to equivalent responses compatible with said first language and communicating said equivalent responses to said outside of said network*. (Emphasis added.)

Clearly, this claim calls for, interalia, (1) receiving first requests from outside of the network in a first language that is incompatible with architecture of a storage system within the network (at least Applicants' block 303, Fig. 3), (2) obtaining responses to the first requests in a second language compatible with the architecture (at least Applicants' block 302, Fig. 3); and (3) converting the responses to equivalent responses that are compatible with the first language (at least Applicants' block 302, Fig. 3) and communicating them to outside of the network (at least Applicants' block 303, Fig. 3). None of this is shown in Eager or Bapat.

Eager shows a conversion from enterprise (legacy) to distributed (non-legacy) infrastructure (Title and Abstract - also, *see* col. 5, lines 9-15). Operation of Eager generates a new system, namely distributed software, but after the translation or transition is made, the enterprise software is out of the picture and no longer interacts with the distributed software. Outside requests formulated in non-legacy distributed software and made of the system after the system converts legacy enterprise to non-legacy distributed architecture simply do not exist in Eager.

What does exist in Eager is a transition between architectures which takes place primarily, if not exclusively, in its re-architecting system 20, Fig. 1 which is shown in

more detail in other Figs. For example, Fig. 16 represents operation within procedural language communication utility block 210 within system 20. Fig. 17 represents operation within user interface conversion utility block 220 within system 20. Fig. 24 represents operation within data definition language conversion utility block 230 within system 20.

Other Figs. show operation outside of re-architecture block 20, but still within the Eager "system" and not from outside thereof. For example, the graphical user interface editor shown in Fig. 28 allows users to work with the distributed software which was generated by transitioning from the enterprise software, the editor being located in block 310 within re-engineering system 30 of Fig. 1. As shown in Fig. 1, the re-engineering system merely interfaces with multi-tiered architecture 10 which resulted from transitioning from enterprise to distributed software. There is no discussion in Eager of continued interaction between enterprise legacy software architecture or information and distributed non-legacy software architecture, after the architecture conversion, as is disclosed and claimed in Applicants' application.

By contrast, Applicants maintain a communication link to outside of the network (*see* at least block 303, Fig. 3) and provide a functionality (*see* at least block 302, Fig. 3) which converts a request from outside of the network formulated in a new, standard language (e.g., XML/CIM) that is incompatible with the legacy language within the network (e.g., RAID++ or C++) to a request that *is* compatible with the network. Then, Applicants reconvert a response to that request, the response being initially formulated in network-compatible language back into the outside-of-the-network language.

Consequently, the Office Action's application of Eager to elements of claim 75 is flawed. On page 16 of the Office Action it states that "receiving first requests from

outside of said network in first language incompatible with said software architecture" is shown by Eager's Fig. 1 and related discussion. But, this isn't true. Not only is there is no discussion in Eager of receiving requests in Fig. 1 from outside of the network, there simply is no discussion in Eager of receiving ANY requests in a first language incompatible with said software architecture, whether one is viewing enterprise architecture or distributed architecture as being analogous to Applicants' "said software architecture". The only possible "requests" shown in Eager may be user requests involving re-engineering which are (1) within the network, and (2) not in a language incompatible with the distributed architecture. Although those user "requests" would be incompatible with the legacy enterprise architecture, there is no reverse operation shown in Eager where the generated distributed architecture communicates with, or converts back to, the legacy enterprise architecture. In other words, Eager simply does not map in any logical or reasonable way to Applicants' claim 75, and Eager is therefore inapplicable to claim 75.

Continuing, however, the Office Action next alleges that Eager, column 2, lines 44-57 shows: "converting said responses [to said first requests] to equivalent responses compatible with said first language and communicating said equivalent responses to said outside of said network." But, this section is merely discussing what is taking place within Eager's Fig. 1 and primarily within its re-architecting system 20, as detailed in Figs. 16, 17 and 24. It has nothing to do with converting responses to be compatible with language from outside of the network. As shown in the previous paragraph, there are no requests from outside of the network, so there cannot be any responses thereto.

The Office Action admits on page 17 that Eager does not disclose "obtaining responses to said first requests in second language compatible with said software architecture" with which Applicants agree. Indeed, there are no first requests at least because there are no "first requests from outside of said network" in Eager.

The Office Action , page 17, then alleges that Bapat discloses "obtaining responses to said first requests in second language compatible with said software architecture" and cites col. 3, lines 53-54; col. 6, lines 28-29; and col. 13, lines 53-54 in support of that position. Again, this is not true, at least because Applicants' "requests" are defined as coming from outside of the network and none of these sections in Bapat deal with requests from outside of Bapat's "network". Moreover, it is noted that each of these sections were previously cited[3] in the Office Action to show subject matter that is unrelated to what they are now being alleged to show, with respect to this "obtaining responses" claim element of claim 75. It is respectfully submitted that this inherent contradiction in applying these same sections of Bapat in two different ways argues against their applicability, at least with respect to claim 75. Therefore, the combination of Eager in view of Bapat does not disclose or suggest the subject matter of the "obtaining responses" step of claim 75.

In view of the above it is respectfully submitted that the 35 U.S.C § 103(a) rejection of claim 75 should be withdrawn and the claim allowed.

Dependent claims 76-80 are dependent from claim 75 and are likewise allowable

[3] Col. 3, lines 53-54, cited to show "means for manipulating" in claim 1; col. 6, lines 28-29 cited to show "to locate public functions and/or data attributes of said header files" in claim 1; col. 13, lines 15-16 cited to show "for emitting code that calls public functions and/or data attributes in the first language in claim 1.

at least for reasons based on their dependency, directly or indirectly, from allowable base claim 75.

On page 16 of the Office Action, claims 63-67 are alleged to correspond to claims 1, 34, 10, 21 and 2 respectively and are rejected under the same rationale set forth in connection with the rejection of those claims. It further alleges that claims 68-69 correspond to claims 1 and 34 and are rejected under the same rationale set forth in connection with the rejection of those claims. Applicants respectfully disagree with these two rejections and submit that claims 63 and 68 clearly track claim 75 more closely than claim 1, because claims 63 and 68 each recites virtually the same three claim elements of receiving first requests, obtaining responses to the first requests, and converting the responses to equivalent responses, which were recited in claim 75, but not in claim 1.

Claims 63 and 68 are therefore allowable for reasons given above with respect to claim 75. And claims 64-67, dependent from claim 63 as well as claim 69, dependent from claim 68 are also allowable, at least for reasons based on their dependencies from allowable base claims 63 or 68. To the extent that the Examiner may insist that claims 63 and 68 correspond to claim 1, then Applicants submit that claims 63 and 68 are allowable, at least for the same reasons given for allowance of claim 1.

35 USC § 102(b):

Claims 70-73, 81-83 and 89 are rejected under 35 USC § 102(b) as being anticipated by Eager. Applicants respectfully traverse this rejection for the following reasons.

Consider, for example, claim 70:

> A method for managing functional systems to be practiced on a computer compatible with computer software architecture comprising: *receiving first*

> *requests* in first language incompatible with said computer software architecture; *obtaining responses to said first requests* in second language compatible with said computer software architecture; and, *converting said responses to equivalent responses* compatible with said first language and communicating said equivalent responses to the destination from which, or to destinations related to that from which, said first requests originated.

It is clear that this claim includes the same three elements as recited in each of claims 63, 68 and 75. As noted above in connection with remarks about these three claims, Eager does not disclose or suggest any one of these three claim elements. The Office Action, page 26 attempts to map certain sections of Eager against Applicants' claim elements of claim 70. This fails because Eager is fundamentally deficient with respect to Applicants' claim 70. Eager does not entertain continued contact with its legacy system after conversion, but Applicants' claimed invention does. In other words, Eager transitions enterprise architecture to distributed architecture, and thereafter it runs in distributed architecture without re-visiting its enterprise architecture. A major difference between Eager and Applicants' invention is that after a transition from legacy software to non-legacy software is obtained, the method of claim 70 recites subject matter that permits a return to its legacy system to obtain responses to inquiries received in the new, non-legacy software.

Consider the Office Action's attempted mapping of sections in Eager to claim 70. To begin with, the Office Action, page 26, cites Eager, col. 4, lines 23-30 to show "A method for managing functional systems to be practiced on a computer compatible with computer software architecture." This section in Eager discusses the operation of Eager's Fig. 1. Applicants' recited "computer software architecture" is legacy architecture. Any legacy architecture in Eager would have to be within block 20, before it is transitioned

into the multi-tiered architecture 10. However, this sets up a contradiction in the Office Action.

The Office Action cites Fig. 22 and related discussion to show "receiving first requests in a first language incompatible with said computer software architecture." The "first language" is Applicants' new, standardized language. But, Fig. 22 shows nothing more than activity within procedural language conversion utility 220 (see Fig. 17) which is all contained within re-architecting system 20 of Fig. 1, Eager's transitioning process. If block 20 is alleged to contain legacy architecture, which is the only plausible conclusion one can draw from column 4, lines 23-30 cite, then it cannot also be used to show "receiving first requests in a first language incompatible with said computer software architecture." In Fig. 22, intermediary language programs (meta language) 225 is not analogous to receiving requests in Applicants' first language. Neither is target language code & data structures 227.

The Office Action then relies on Figs. 22 and 24 and related discussion to show: "obtaining responses to said first requests in second language compatible with said computer software architecture." Again, Figs. 22 and 24 show operation within data definition language utility 230 within block 20 of Fig. 1. "Second language compatible with said computer software architecture" is the legacy language. Thus, Figs. 22 and 24 have absolutely nothing to do with obtaining responses in old legacy language to requests made in a new language and they are limited to only showing a transitioning from legacy enterprise to non-legacy distributed software.

The Office Action then relies on col. 2, lines 44-57 to show "converting said responses to equivalent responses compatible with said first language and communicating

said equivalent responses to the destinations from which, or to destinations related to that from which, said first requests originated." But, this section merely discusses nothing more than transitioning from source applications into new target applications and says absolutely nothing about converting responses made in legacy software back into equivalent responses made in the new, standard software for communication back to those destinations which originated the requests.

As previously mentioned, the equivalent of Applicants' blocks 303 and 302 of Fig. 3, which pictorially represent the functioning of at least some of the subject matter of claim 70, do not exist in Eager.

Therefore, Eager does not disclose or suggest claim 70 and it is respectfully submitted that the 35 U.S.C. § 102(b) of claim 70 be withdrawn and the claim allowed.

Claims 71-74 are dependent from claim 70 and are allowable at least for reasons based on their dependency, directly or indirectly, from an allowable base claim.

Finally, independent claim 81 recites, interalia, "an interface between said first computer network and said second computer network to automatically **convert** communication **from said second** computer network **into** a form compatible with **said first** computer network, **and** to automatically **convert response** to said communication generated **by said first** computer network **into** a form compatible with **said second** computer network." It is clear that this claim recites automatic conversion of communication in TWO directions, from said second to said first and from said first to said second, i.e., both directions. It is also clear that, at best, Eager shows a translation or conversion from legacy to non-legacy software architecture in only ONE direction. There is no disclosure in Eager suggesting a return to the legacy architecture after

conversion, much less a disclosure in Eager of any automatic conversion of communication from legacy to non-legacy AND vice versa. Accordingly, Eager does not disclose or suggest claim 81 and it is respectfully submitted that the 35 U.S.C. § 102(b) of claim 81 be withdrawn and the claim allowed.

Claims 82-89 are dependent from claim 81 and are allowable at least for reasons based on their dependency, directly or indirectly, from an allowable base claim.

CONCLUSION

In view of the above arguments, the rejection of claims 1-89 should be withdrawn. Reconsideration and allowance are respectfully requested.

To the extent that the above-discussed, or any other, Office Action citations of Eager, "admitted prior art", and/or Bapat were applied against particular dependent claim elements but not expressly rebutted herein, it is to be understood that Applicants' silence does not mean or imply acquiescence. Rather, Applicants believe that any response to application of such citations would be moot in view of the foregoing arguments and provisions of MPEP §§ 2131 and 2143.

To the extent that an extension of time may be needed in order to enter this amendment in this case, please consider this response as including a petition under 37 C.F.R. § 1.136 for such extension of time. Please charge any fee for such petition or any other fee or cost that may be incurred by way of this amendment to Patent Office deposit account number **05-0889**.

If the Examiner feels that a telephone conversation may serve to advance the prosecution of this application, he or she is invited to telephone Applicants' undersigned representative at the telephone number provided below.

Respectfully submitted,

JOEL WALL
Registration No. 25,648

TELEPHONE:
JOEL WALL ESQ.
508-625-1323
September 19, 2005